Lab Report #7

Hanz Chua U26738740
Viraj Amarasinghe U14806439
Jian Gong U42910460

Introduction

The purpose of this lab was to help us gain a deeper understanding of sequential circuit design, and in particular, counters. We designed and implemented a 3-bit gray code counter using JKFFs.

Methods and Materials

Wires
Breadboard
2 JK Flip Flop ICs
2 AND Gate ICs
1 OR Gate IC
3 LED
3 Resistors
Push Button

In this lab, we designed a gray code counter. We began with drawing the state transition diagram and next state table for the gray code counter. We designed the circuit and implemented it using JK flip flops, since we did not have D or T flip flops. We set up the function generator to produce a 1 Hz square wave. We made the signal slow enough so we could see the changing states using LEDs. We then implemented the circuit and tested its functionality

Results

We observed that the gray code counter circuit worked as planned; When we pressed the button the LEDs lit up to the corresponding number and sequence associated with the grey code order. We also observed that we can also observe the output after connecting the circuit to a 7-segment display rather than the LEDs. After resolving some issues/mistakes, we concluded that the results came out as expected and contributed to the goal of the experiment; to help us gain a deeper understanding of sequential circuit design and counters.

Conclusion

In conclusion, when someone presses the button on our circuit, the counter counts up and the output, whether it is an LED or 7 segment display, updates to the corresponding state. Issues we ran into included omitting resistors, burning out LEDs and even a display, and plugging in wires to wrong holes.

Questions

1. Verilog testbench, module, and EPWave

Testbench

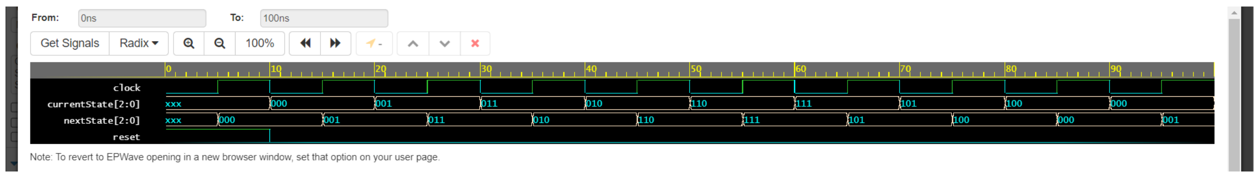
```
module testbench();
  reg clock, reset;
  reg [2:0] currentState;
  wire [2:0] nextState;

  gray_counter counter(clock, reset, currentState, nextState);

  initial begin
    clock = 0;
    reset = 1;
    forever #5 clock = ~clock;
  end

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;

    #10 reset = 0;
    currentState = 3'b000;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);
    currentState = nextState;

    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);

    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);

    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);

    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);

    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);
```

```
    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);

    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);

    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);

    #10 currentState = nextState;
    $monitor("Clock: %b, Current state = %b, Next state = %b\n",
clock, currentState, nextState);

    $finish;
  end
Endmodule
```

Module

```
module gray_counter(clock, reset, currentState, nextState);
  input clock, reset;
  input reg [2:0] currentState;
  output reg[2:0] nextState;
  always @(posedge clock) begin
    if (reset) begin
      nextState <= 3'b000;
    end
    else begin
      case(currentState)
        (3'b000) : nextState <= 3'b001;
        (3'b001) : nextState <= 3'b011;
        (3'b011) : nextState <= 3'b010;
        (3'b010) : nextState <= 3'b110;
        (3'b110) : nextState <= 3'b111;
        (3'b111) : nextState <= 3'b101;
        (3'b101) : nextState <= 3'b100;
        (3'b100) : nextState <= 3'b000;
      endcase
    end
  end
endmodule
```

EPWave



From: 0ns To: 100ns

```
                   0         10        20        30        40        50        60        70        80        90
clock
currentState[2:0]  xxx        000       001       011       010       110       111       101       100       000
nextState[2:0]     xxx   000       001       011       010       110       111       101       100       000       001
reset
```

Note: To revert to EPWave opening in a new browser window, set that option on your user page.

2. State transition diagram, truth tables, and K-Maps

$$\begin{tikzcd}
 & 100 \arrow[r] & 000 \arrow[r] & 001 \arrow[d] \\
 & 101 \arrow[u] & & 011 \arrow[d] \\
 & 111 \arrow[u] & 110 \arrow[l] & 010 \arrow[l]
\end{tikzcd}$$

T_1	T_2	T_3
0	0	1
0	1	0
1	0	0
0	0	1
1	0	0
0	0	1
0	0	1
0	1	0

x	y	z	x^+	y^+	z^+	J_x	K_x	J_y	K_y	J_z	K_z
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	1	0	x	1	x	x	0
0	1	0	1	1	0	1	x	x	0	0	x
0	1	1	0	1	0	0	x	x	0	x	1
1	0	0	0	0	0	x	1	0	x	0	x
1	0	1	1	0	0	x	0	0	x	x	1
1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	1	0	1	x	0	x	1	x	0

yz \ x	0	1
00	0	0
01	0	1
11	0	1
10	1	1

$x^+ = xz + y\bar{z}$

yz \ x	0	1
00	0	0
01	1	0
11	1	0
10	1	1

$y^+ = \bar{x}z + y\bar{z}$

yz \ x	0	1
00	1	0
01	1	0
11	0	1
10	0	1

$z^+ = \bar{x}\bar{y} + xy$

yz \ x	0	1
00	0	x
01	0	x
11	0	x
10	1	x

$J_x = y\bar{z}$

yz \ x	0	1
00	x	1
01	x	0
11	x	0
10	x	0

$K_x = \bar{y}\bar{z}$

yz \ x	0	1
00	0	0
01	1	0
11	x	x
10	x	x

$J_y = \bar{x}z$

yz \ x	0	1
00	x	x
01	x	x
11	0	0
10	0	1

$K_y = x\bar{z}$

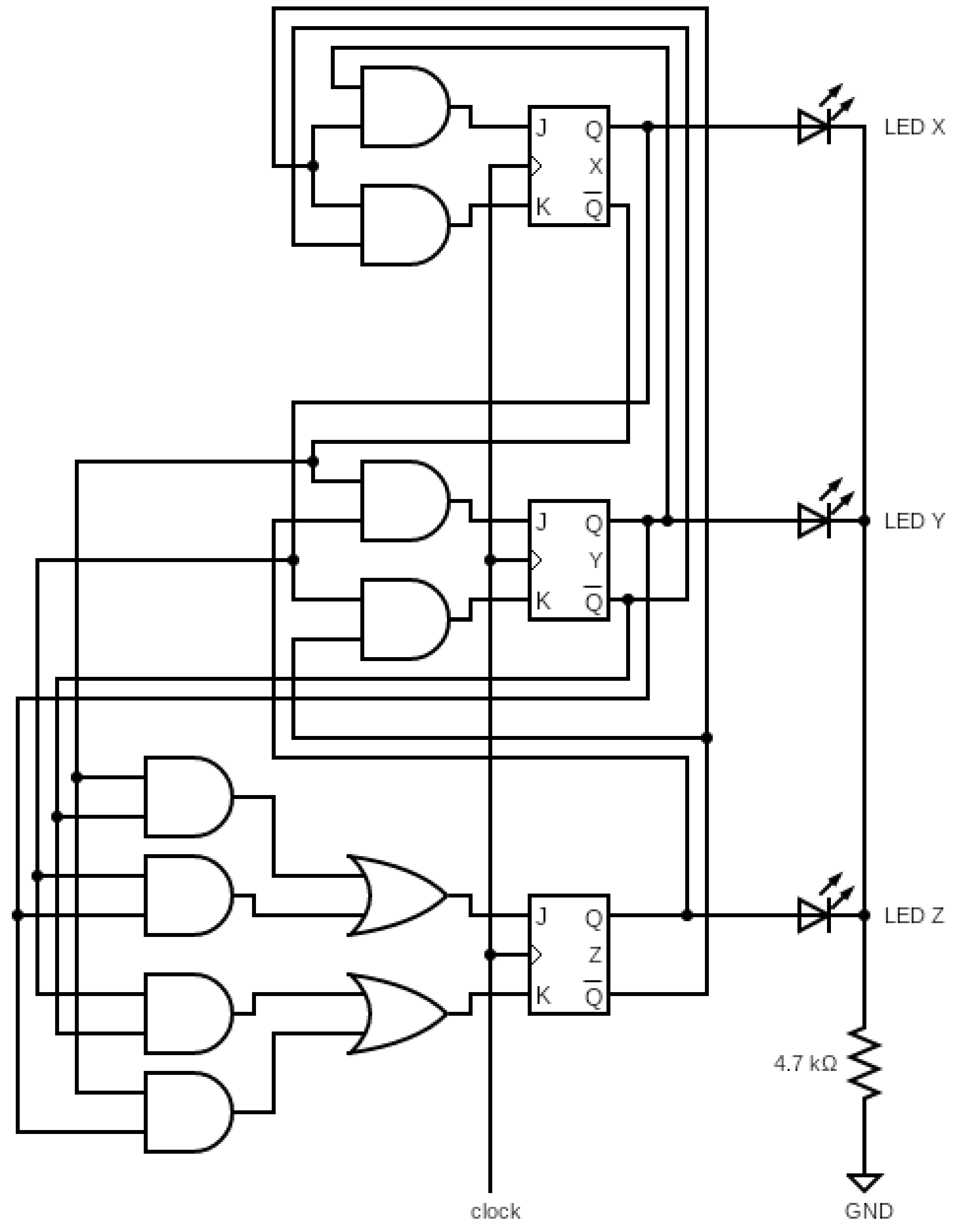
yz \ x	0	1
00	1	0
01	x	x
11	x	x
10	0	1

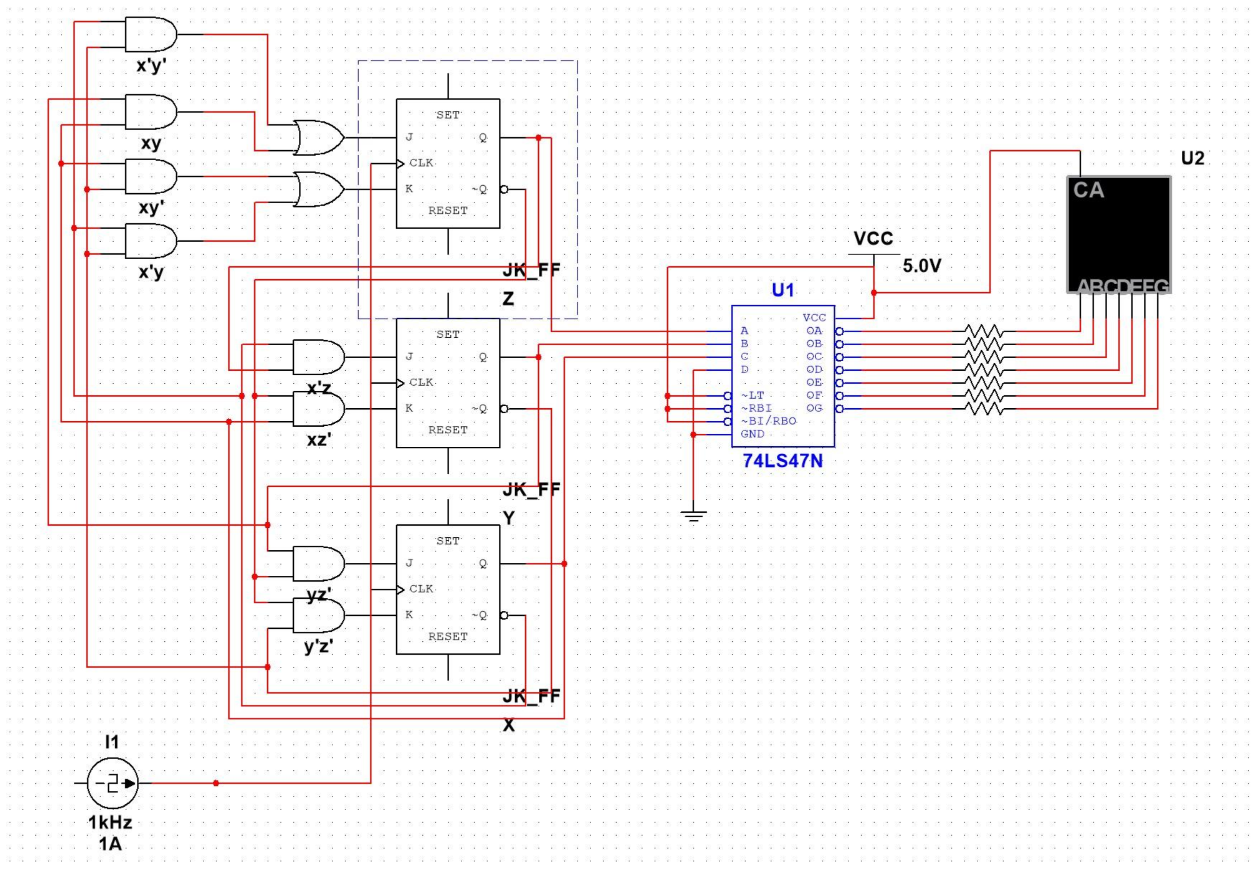
$J_z = \bar{x}\bar{y} + xy$

yz \ x	0	1
00	x	x
01	0	1
11	1	0
10	x	x

$K_z = x\bar{y} + \bar{x}y$

3. Circuit Diagrams





x'y'
xy
xy'
x'y
SET
J
Q
CLK
K
~Q
RESET
JK_FF
Z
x'z
xz'
JK_FF
Y
yz'
y'z'
JK_FF
X
I1
1kHz
1A
VCC
5.0V
U1
VCC
A
B
C
D
~LT
~RBI
~BI/RBO
GND
OA
OB
OC
OD
OE
OF
OG
74LS47N
U2
CA
ABCDEFG